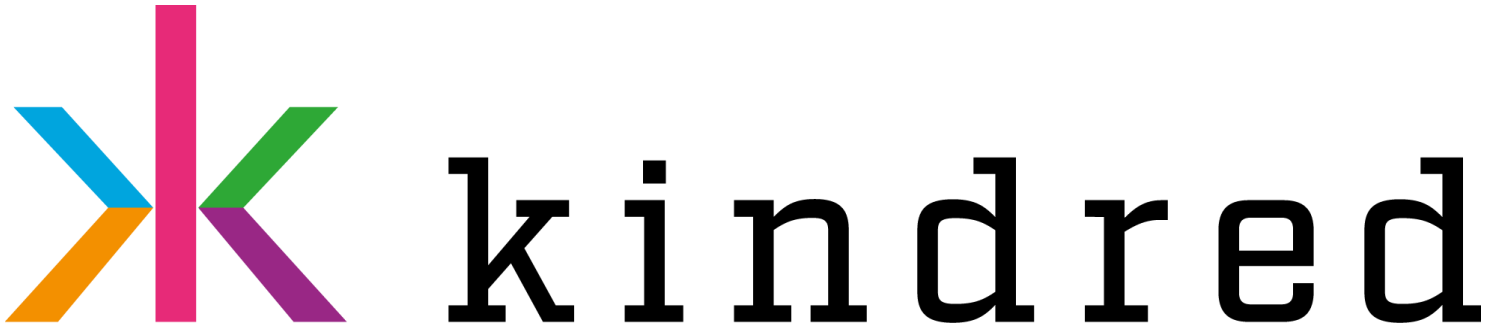
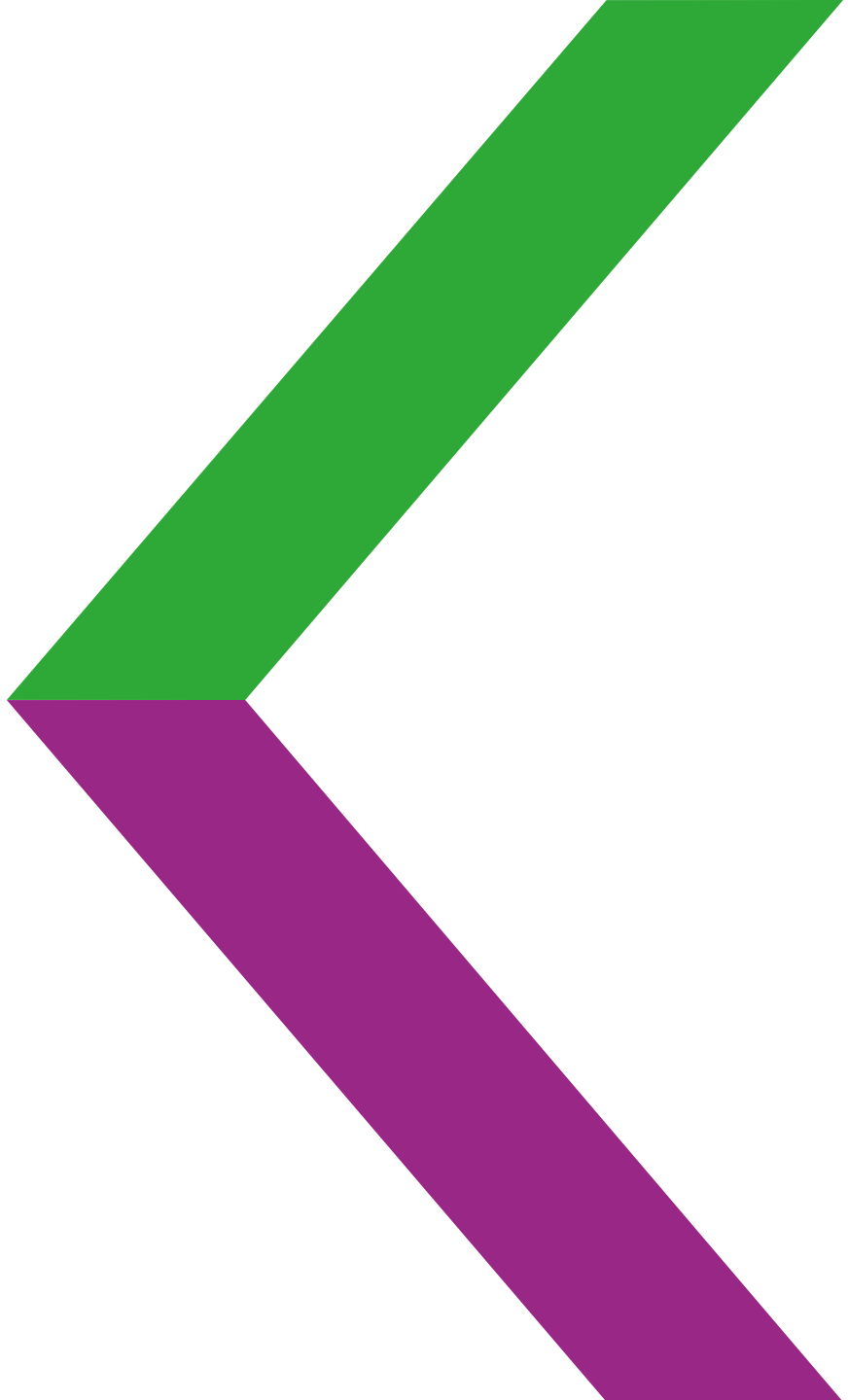


Nolla Utanförskapet: FE React

Kindred United

Agenda

- React
- Examples
- Exercises
- Useful links
- Questions?
- Home assignment



© Kindred Group plc 2018

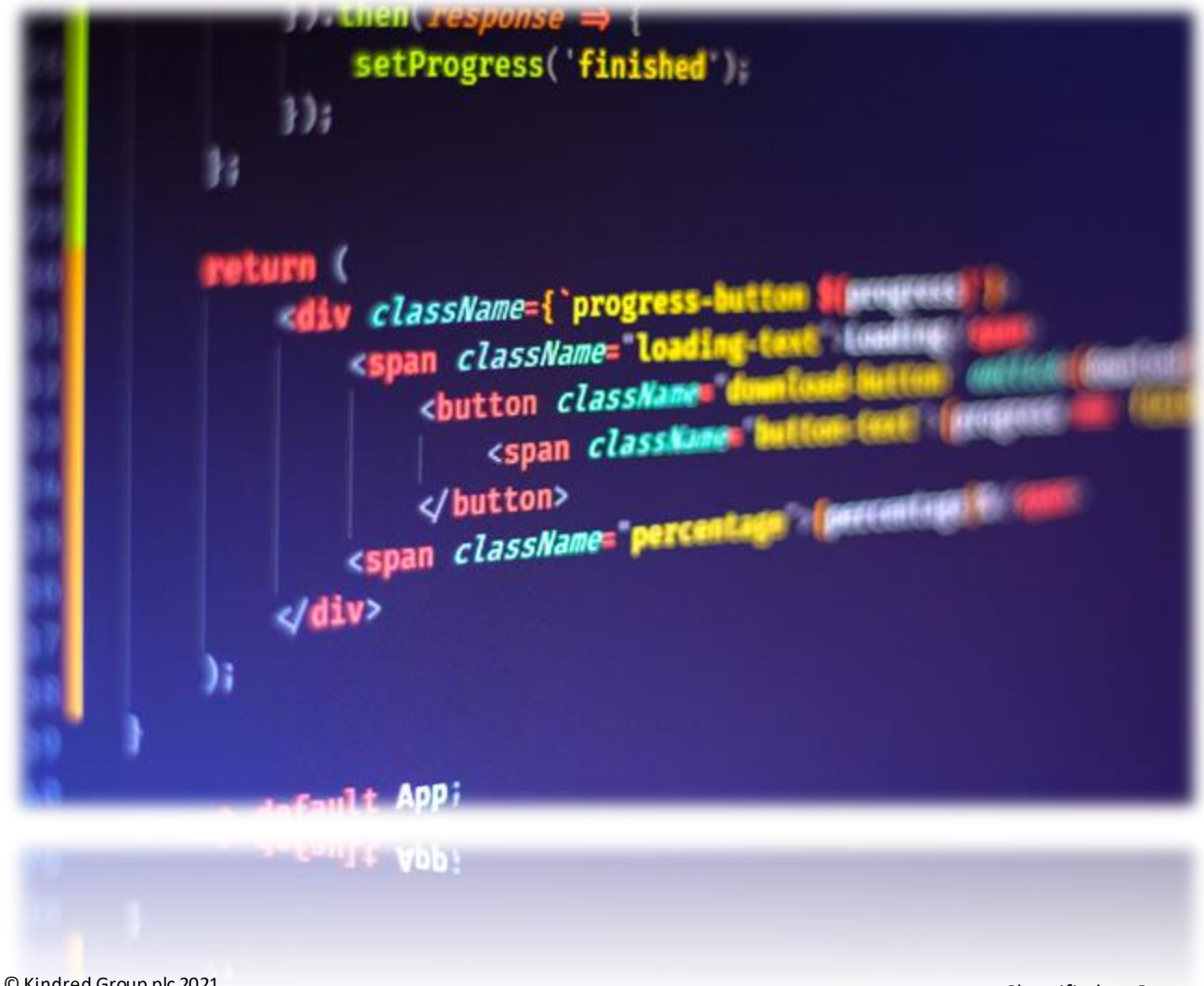
Course Teachers

HTML + CSS: (3rd February 2022)	• Elena Insheva • Ishaan Puniani
JavaScript: (8th February 2022)	• Elena Insheva • Erika Boberg
React: (22nd February 2022)	• Elena Insheva • Harshit Jyoti

© Kindred Group plc 2021

Classified as General

React



React is a JavaScript library for building user interfaces.

React is used to build single-page applications.

React allows us to create reusable UI components.

© Kindred Group plc 2021

Classified as General

1st React Example

```
import React from "react";
import ReactDOM from "react-dom";

function Hello(props) {
  return <h1>Hello World!</h1>;
}

ReactDOM.render(<Hello />, document.getElementById("root"));

/*
The examples in this tutorial are created using the
create-react-app.

Install the create-react-app,
and you will be able to run the same examples on your computer.

In this example we create a component called 'Hello'.
The component is rendered in a container called 'root'.
*/
```

© Kindred Group plc 2021

Classified as General

Setting up a React Environment

Install VS code >> https://code.visualstudio.com/download

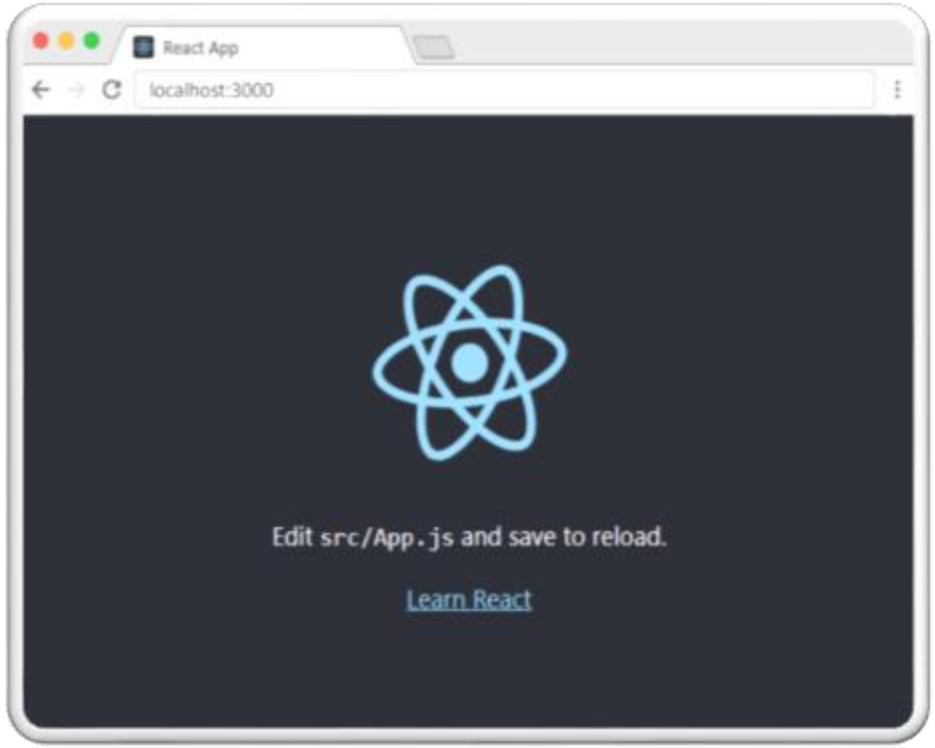
Install Node >> https://nodejs.org/en/download/

Install npm >> run the command in your VS terminal : **npm install -g npm**

Create your React app >> **npx create-react-app my-first-nolla-app**

Open the folder with new app >> **cd my-nolla-app**

Let's start/run the project >> **npm start** >> a new browser window will pop up with your newly created React App! If not, open your browser and type localhost:3000 in the **browser** address bar.



© Kindred Group plc 2021

Classified as General

Modify the React Application

Look in the "**my-nolla-app**" directory, and you will find a "**src**" folder.

Inside the "src" folder there is a file called "App.js", open it and it will look like this >>

Try changing the HTML content and save the file.

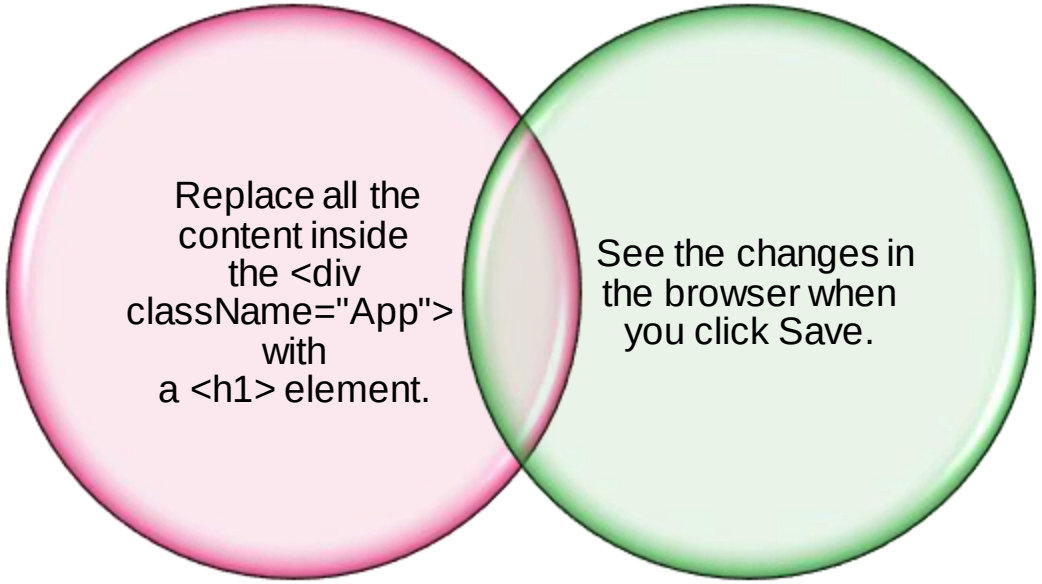
Notice that the changes are visible immediately after you save the file, you do not have to reload the browser!

```
import logo from './logo.svg';
import './App.css';

function App() {
  return (
    <div className="App">
      <header className="App-header">
        <img src={logo} className="App-logo" alt="logo" />
        <p>
          Edit <code>src/App.js</code> and save to reload.
        </p>
        <a
          className="App-link"
          href="https://reactjs.org"
          target="_blank"
          rel="noopener noreferrer"
        >
          Learn React
        </a>
      </header>
    </div>
  );
}

export default App;
```

© Kindred Group plc 2021

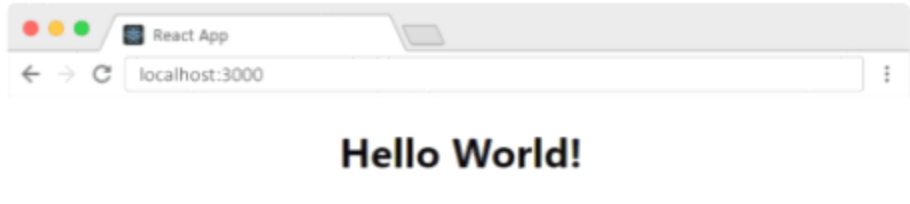
Classified as General

Modify the React Application




```
function App() {
  return (
    <div className="App">
      <h1>Hello World!</h1>
    </div>
  );
}

export default App;
```

The result:



© Kindred Group plc 2021

Classified as General

Clean-up the React Application

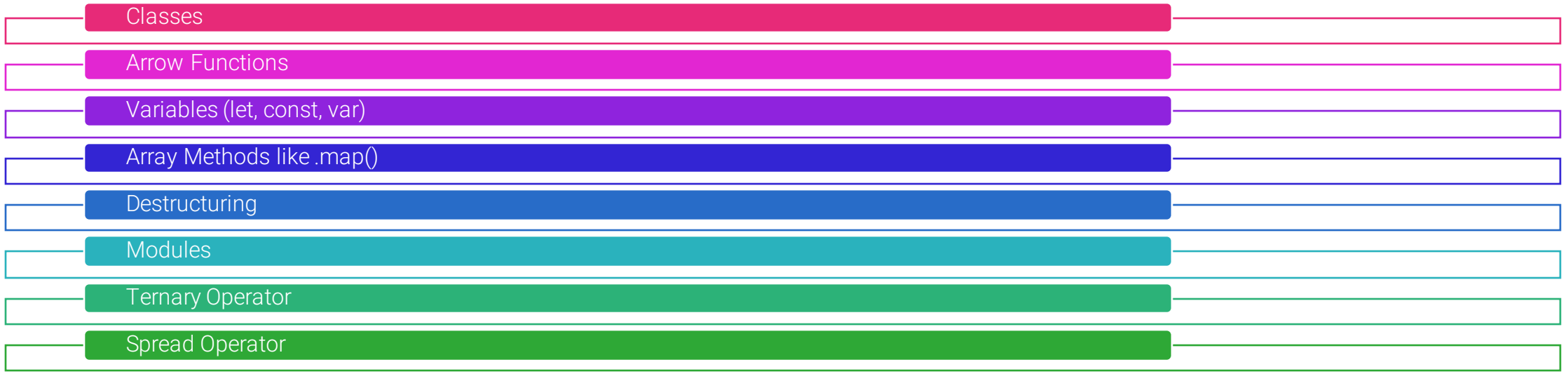
Strip down the **"src"** folder to only contain one file: "index.js"

Remove any unnecessary lines of code inside the "index.js"

© Kindred Group plc 2021

React ES6 (standardized JavaScript v.6)

- React uses ES6 and you should be familiar with some of the new features like:



- Note: it's a self-learning part more info here: https://www.w3schools.com/react/react_es6.asp
- Go into every example and try to play with it, changing data and observing the result

© Kindred Group plc 2021

Classified as General

React ES6 Classes

A class is a type of function, but instead of using the keyword function to initiate it

We use the keyword class >> class name begins with uppercase

Properties are assigned inside a constructor() method.

You can create objects using the Car class:

Example

A simple class constructor:

```
class Car {
  constructor(name) {
    this.brand = name;
  }
}
```

```
<!DOCTYPE html>
<html>

<body>

<script>
class Car {
  constructor(name) {
    this.brand = name;
  }
}

const mycar = new Car("Ford");

document.write(mycar.brand);
</script>

</body>
</html>
```

Ford

© Kindred Group plc 2021

Methods in ES6 Classes

You can add your own methods in a class >> create a method named "present"

```
class Car {
  constructor(name) {
    this.brand = name;
  }

  present() {
    return 'I have a ' + this.brand;
  }
}

const mycar = new Car("Ford");
mycar.present();
```

```
<!DOCTYPE html>
<html>

<body>

<script>
class Car {
  constructor(name) {
    this.brand = name;
  }

  present() {
    return 'I have a ' + this.brand;
  }
}

const mycar = new Car("Ford");
document.write(mycar.present());
</script>

</body>
</html>
```

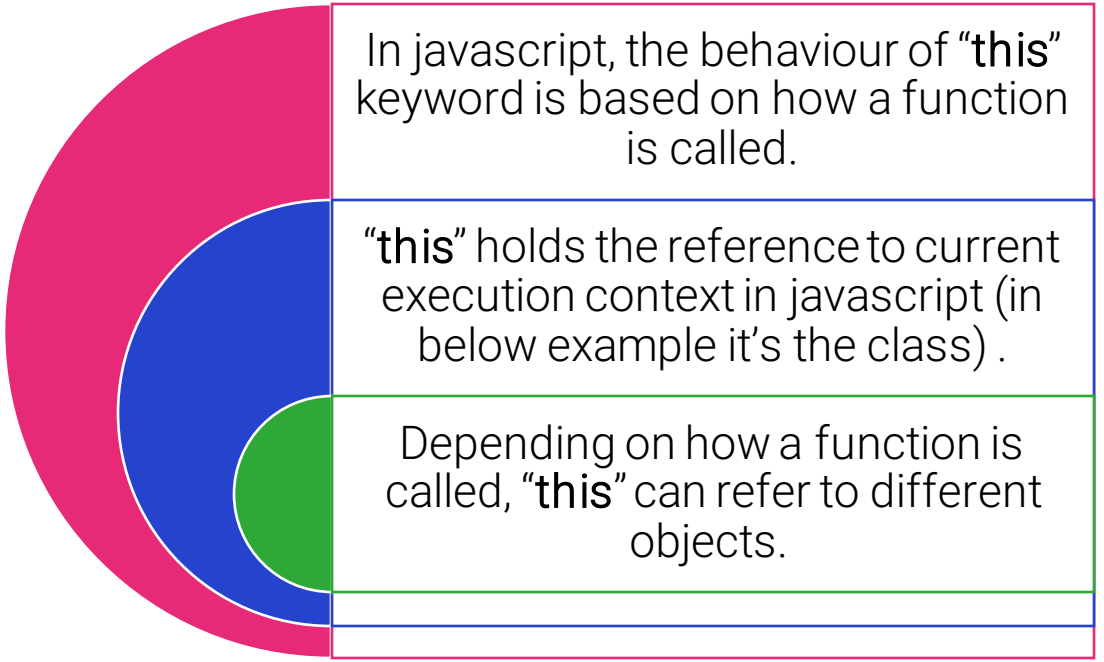
I have a Ford

© Kindred Group plc 2021

Classified as General

this?

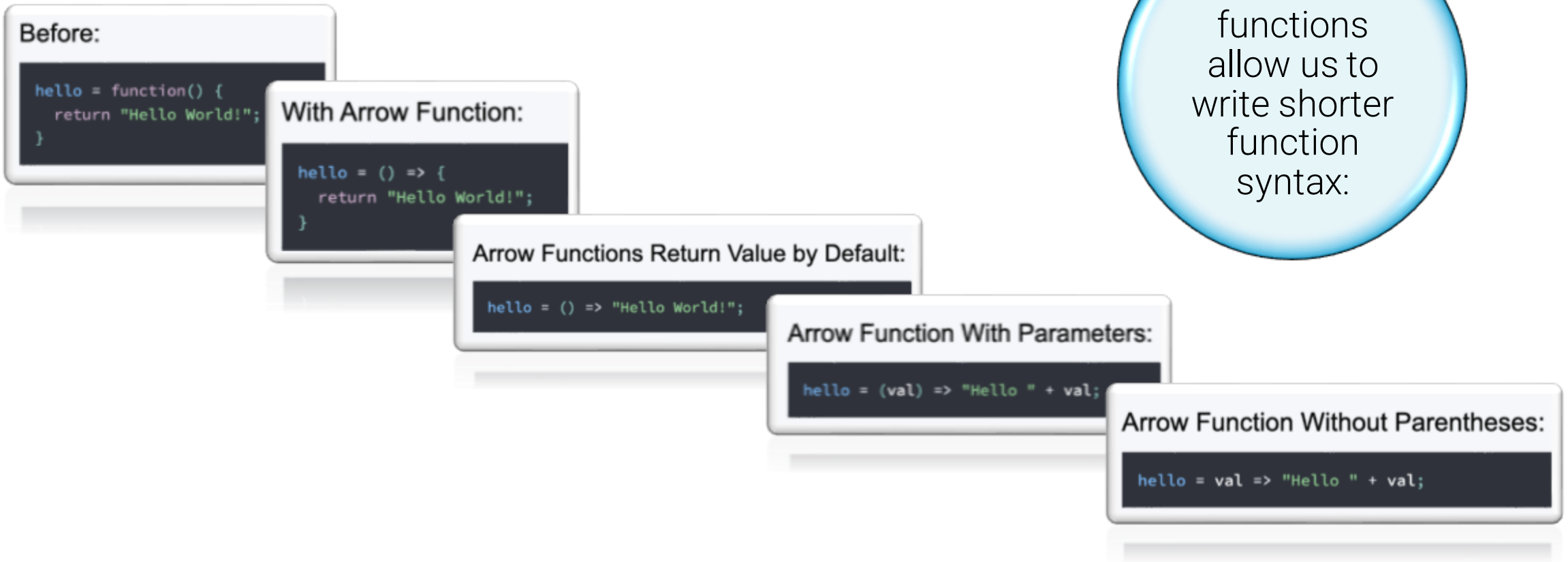
```
import React, { Component } from 'react';

class App extends Component {
  constructor(props) {
    super(props);
    this.clickFunction = this.clickFunction.bind(this);
  }

  clickFunction() {
    console.log(this.props.value);
  }

  render() {
    return <div onClick={this.clickFunction}>Click Me!</div>;
  }
}

export default App;
```


In javascript, the behaviour of “this” keyword is based on how a function is called.
“this” holds the reference to current execution context in javascript (in below example it’s the class) .
Depending on how a function is called, “this” can refer to different objects.

© Kindred Group plc 2021

Classified as General

React ES6 Arrow Functions


Before:
hello = function() {
return "Hello World!";
}
With Arrow Function:
hello = () => {
return "Hello World!";
}
Arrow Functions Return Value by Default:
hello = () => "Hello World!";
Arrow Function With Parameters:
hello = (val) => "Hello " + val;
Arrow Function Without Parentheses:
hello = val => "Hello " + val;

Arrow functions allow us to write shorter function syntax:

© Kindred Group plc 2021

Classified as General

React ES6 Modules

Modules

- JavaScript modules allow you to break up your code into separate files.
- This makes it easier to maintain the code-base.
- ES Modules rely on the import and export statements.

Export

- You can export a function or variable from any file.
- Let us create a file named person.js, and fill it with the things we want to export.
- There are two types of exports: Named and Default.

Import

- You can import modules into a file in two ways, based on if they are named exports or default exports.
- Named exports must be destructured using curly braces. Default exports do not.

```
import React from 'react';
import { Address, PhoneNumber, Email } from '../fieldsets';
import { EditUserDetailsHeader as Header } from '../index';
import styles from '../common/WidthAndMargins.style';

const UserDetails = () => {
  return (
    <section style={styles.section}>
      <Header />
      <Address />
      <PhoneNumber />
      <Email />
    </section>
  );
};

export default UserDetails;
```

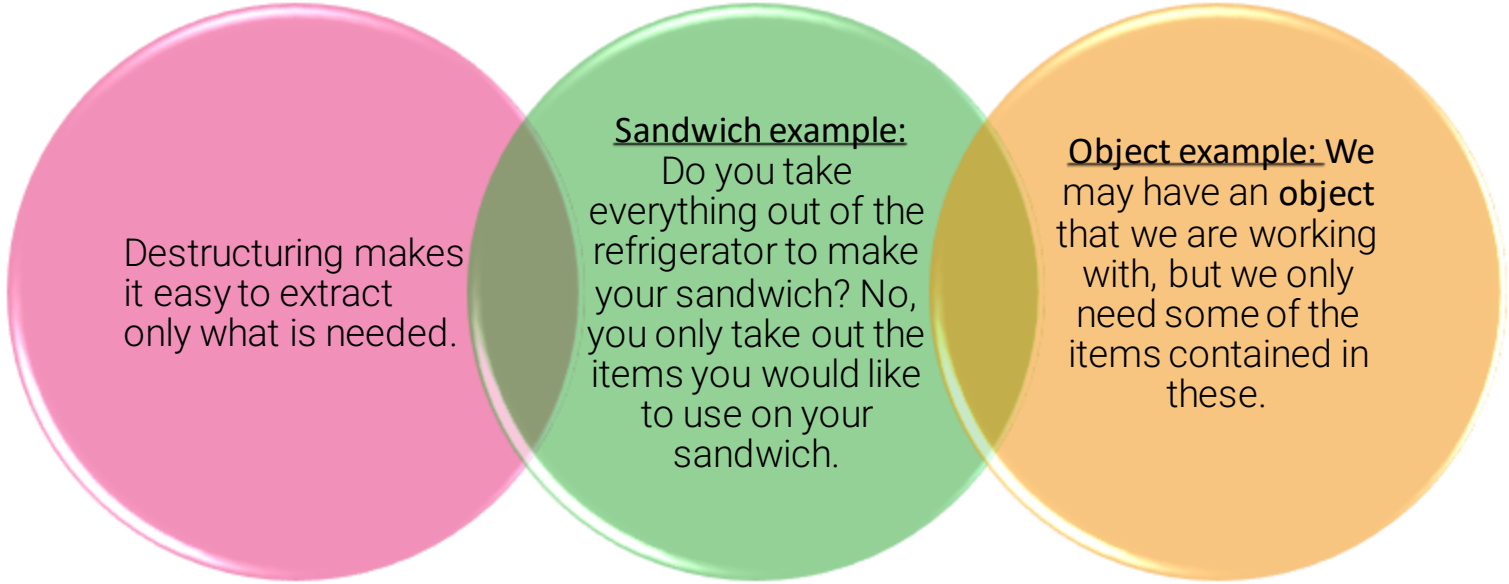
```
import React from 'react';
import { spacing } from 'kindred-ui-kit/foundation';

const Header = () => {
  const title = 'My title';
  const subTitle = 'My subtitle';

  return (
    <div style={{ marginBottom: spacing.xl }}>
      {title}
      <div>{subTitle}</div>
    </div>
  );
};

export default Header;
```

```
import React from 'react';
import { useTranslation } from 'kmf-i18n';
import { Grid, Box, HorizontalRule } from 'kindred-ui-kit/primitives';
import { useStateContext } from '../../machine/StateContext';
import styles from '../common/WidthAndMargins.style';
import Header from '../Header/Header.component';
import Spinner from '../common/Spinner/Spinner.component';
import EditButton from './EditButton/EditButton.component';
import PersonalDetails from './PersonalDetails.component';
```

© Kindred Group plc 2021

Classified as General

React ES6 Destructuring


Destructuring makes it easy to extract only what is needed.
Sandwich example: Do you take everything out of the refrigerator to make your sandwich? No, you only take out the items you would like to use on your sandwich.
Object example: We may have an object that we are working with, but we only need some of the items contained in these.

Before:

```
const vehicleOne = {
  brand: 'Ford',
  model: 'Mustang',
  type: 'car',
  year: 2021,
  color: 'red'
}

myVehicle(vehicleOne);

// old way
function myVehicle(vehicle) {
  const message = 'My ' + vehicle.type + ' is a ' + vehicle.color + ' ' + vehicle.brand + ' ' + vehicl
}
```

With destructuring:

```
const vehicleOne = {
  brand: 'Ford',
  model: 'Mustang',
  type: 'car',
  year: 2021,
  color: 'red'
}

myVehicle(vehicleOne);

function myVehicle({type, color, brand, model}) {
  const message = 'My ' + type + ' is a ' + color + ' ' + brand + ' ' + model + '.';
}
```

© Kindred Group plc 2021

Classified as General

React Render HTML

React's goal is in many ways to render HTML in a web page.

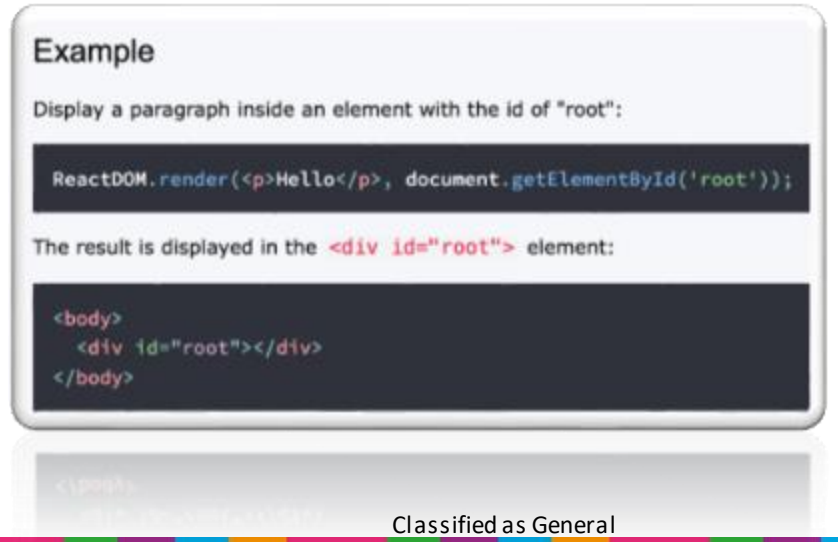
React renders HTML to the web page by using a function called ReactDOM.render().

The ReactDOM.render() function takes two arguments, HTML code and an HTML element.

The purpose of the function is to display the specified HTML code inside the specified HTML element.

But render where? There is another folder in the root directory of your React project, named "public". In this folder, there is an index.html file.

You'll notice a single in the body of this file. This is where our React application will be rendered.


Example
Display a paragraph inside an element with the id of "root":
ReactDOM.render(Hello , document.getElementById('root'));
The result is displayed in the element:
<body>

</body>

© Kindred Group plc 2021

Classified as General

React JSX

- JSX stands for JavaScript XML.
- JSX allows us to write HTML in React.
- JSX makes it easier to write and add HTML in React.
- JSX allows us to write HTML elements in JavaScript and place them in the DOM without any createElement() and/or appendChild() methods.
- JSX converts HTML tags into react elements.

Example 1

JSX:

```
const myelement = <h1>I Love JSX!</h1>;

ReactDOM.render(myelement, document.getElementById('root'));
```

Example 2

Without JSX:

```
const myelement = React.createElement('h1', {}, 'I do not use JSX!');

ReactDOM.render(myelement, document.getElementById('root'));
```

© Kindred Group plc 2021

Classified as General

JSX rules

Expressions in JSX
With JSX you can write expressions inside curly braces {}

Example

Execute the expression 5 + 5:

```
const myelement = <h1>React is {5 + 5} times better with JSX</h1>;
```

Inserting a Large Block of HTML
To write HTML on multiple lines, put the HTML inside parentheses:

Example

Create a list with three list items:

```
const myelement = (
  <ul>
    <li>Apples</li>
    <li>Bananas</li>
    <li>Cherries</li>
  </ul>
);
```

One Top Level Element
The HTML code must be wrapped in *ONE* top level element.

Example

Wrap two paragraphs inside one DIV element:

```
const myelement = (
  <div>
    <p>I am a paragraph.</p>
    <p>I am a paragraph too.</p>
  </div>
);
```

Elements Must be Closed
JSX follows XML rules, and therefore HTML elements must be properly closed.

Example

Close empty elements with />

```
const myelement = <input type="text" />;
```

© Kindred Group plc 2021

Classified as General

JSX Conditions - if statements

React supports if statements, but not *inside* JSX.

To be able to use conditional statements in JSX, you should put the if statements outside of the JSX, or you could use a ternary expression instead

Example

Write "Hello" if x is less than 10, otherwise "Goodbye":

```
const x = 5;
let text = "Goodbye";
if (x < 10) {
  text = "Hello";
}

const myelement = <h1>{text}</h1>;
```

Example

Write "Hello" if x is less than 10, otherwise "Goodbye":

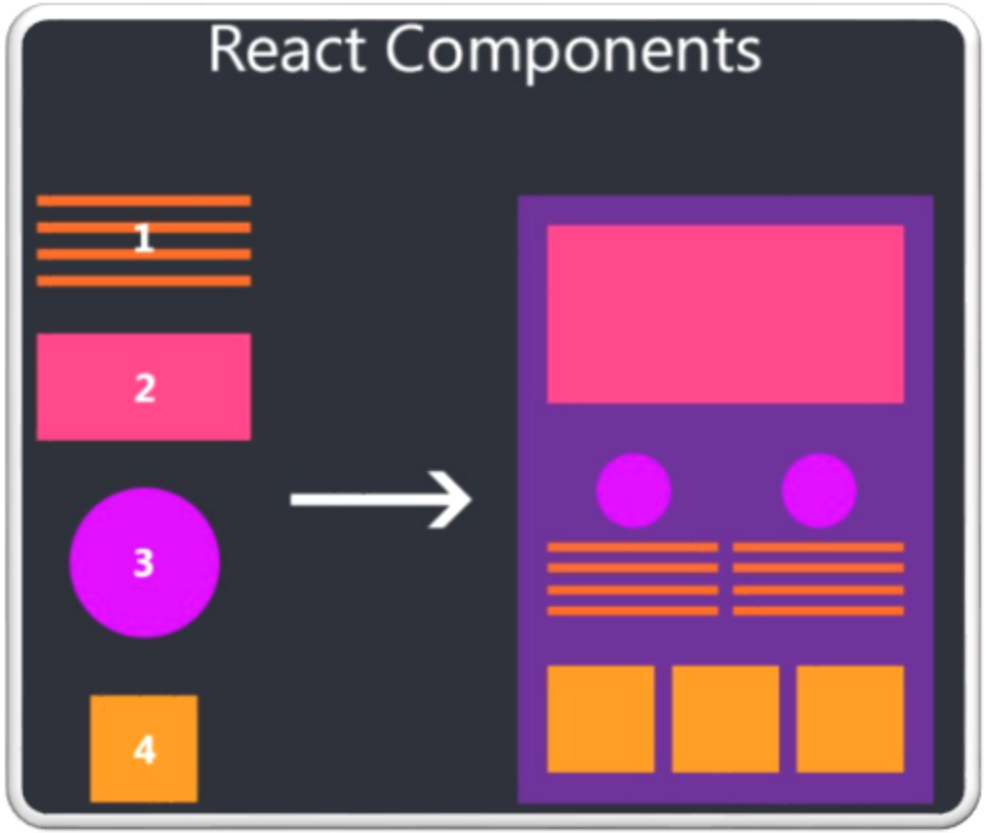
```
const x = 5;

const myelement = <h1>{(x) < 10 ? "Hello" : "Goodbye"}</h1>;
```

© Kindred Group plc 2021

Classified as General

React Components

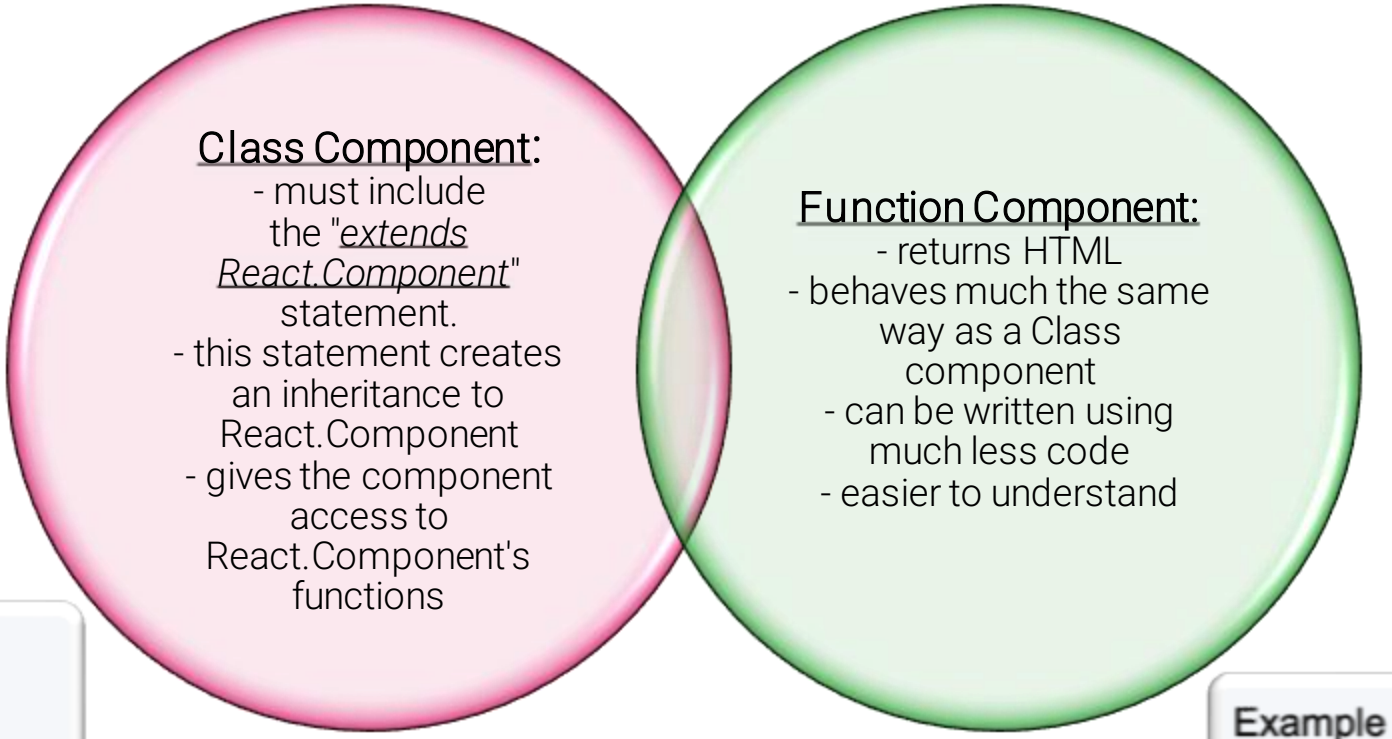
'Display number' component
'Display screen' component
Entire calculator can be a component that wraps around all other sub-components
Each button can be a React component



© Kindred Group plc 2021

Classified as General

React Components: Class VS Function


Class Component:
- must include the "extends React.Component" statement.
- this statement creates an inheritance to React.Component
- gives the component access to React.Component's functions
Function Component:
- returns HTML
- behaves much the same way as a Class component
- can be written using much less code
- easier to understand

Example

Create a Class component called `Car`

```
class Car extends React.Component {
  render() {
    return <h2>Hi, I am a Car!</h2>;
  }
}
```

Example

Create a Function component called `Car`

```
function Car() {
  return <h2>Hi, I am a Car!</h2>;
}
```

© Kindred Group plc 2021

Classified as General

React Components: Props

Components can be passed as props, which stands for properties.

Props are like function arguments, and you send them into the component as attributes.

Props are arguments passed into React components.

Props are passed to components via HTML attributes.

Example

Use an attribute to pass a color to the Car component, and use it in the render() function:

```
function Car(props) {
  return <h2>I am a {props.color} Car!</h2>;
}

ReactDOM.render(<Car color="red"/>, document.getElementById('root'));
```

Example

Add a "brand" attribute to the Car element:

```
const myelement = <Car brand="Ford" />;
```

The component receives the argument as a `props` object:

Example

Use the brand attribute in the component:

```
function Car(props) {
  return <h2>I am a { props.brand }!</h2>;
}
```

© Kindred Group plc 2021

Classified as General

Components in Components

Example

Use the Car component inside the Garage component:

```
function Car() {
  return <h2>I am a Car!</h2>;
}

function Garage() {
  return (
    <>
      <h1>Who lives in my Garage?</h1>
      <Car />
    </>
  );
}

ReactDOM.render(<Garage />, document.getElementById('root'));
```

© Kindred Group plc 2021

Classified as General

Components in Files

Example

This is the new file, we named it "Car.js":

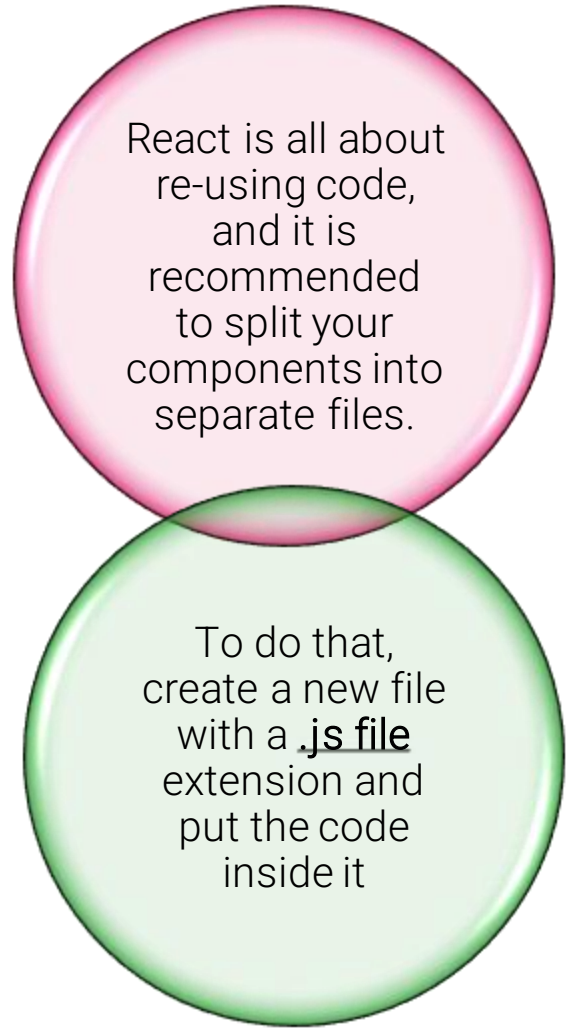
```
function Car() {
  return <h2>Hi, I am a Car!</h2>;
}

export default Car;
```

To be able to use the Car component, you have to import the file in your application.

Example

Now we import the "Car.js" file in the application, and we can use the `Car` component as if it was created here.

```
import React from 'react';
import ReactDOM from 'react-dom';
import Car from './Car.js';

ReactDOM.render(<Car />, document.getElementById('root'));
```


React is all about re-using code, and it is recommended to split your components into separate files.
To do that, create a new file with a .js file extension and put the code inside it

© Kindred Group plc 2021
Classified as General

React Events : adding events

Just like HTML DOM events, React can perform actions based on user events.

React has the same events as HTML: **click, change, mouseover etc.**

React events are written in camelCase syntax: **onClick instead of onclick.**

React event handlers are written inside curly braces: **onClick={shoot} instead of onClick="shoot()"**

React:

```
<button onClick={shoot}>Take the Shot!</button>
```

HTML:

```
<button onclick="shoot()">Take the Shot!</button>
```

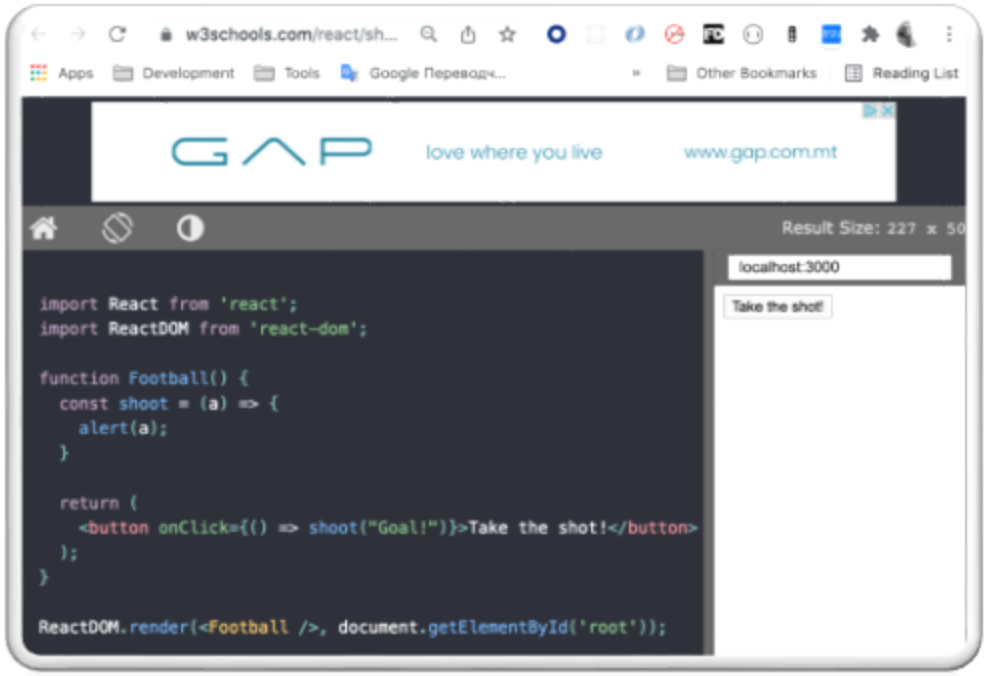
Example:

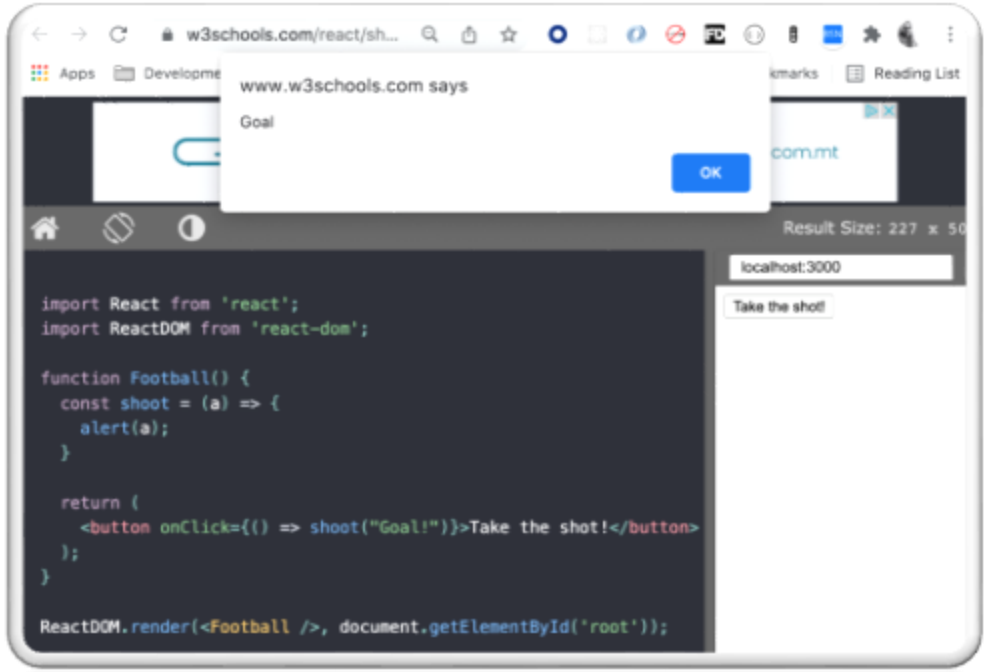
Put the `shoot` function inside the `Football` component:

```
function Football() {
  const shoot = () => {
    alert("Great Shot!");
  }

  return (
    <button onClick={shoot}>Take the shot!</button>
  );
}

ReactDOM.render(<Football />, document.getElementById('root'));
```

© Kindred Group plc 2021

React Events : passing arguments





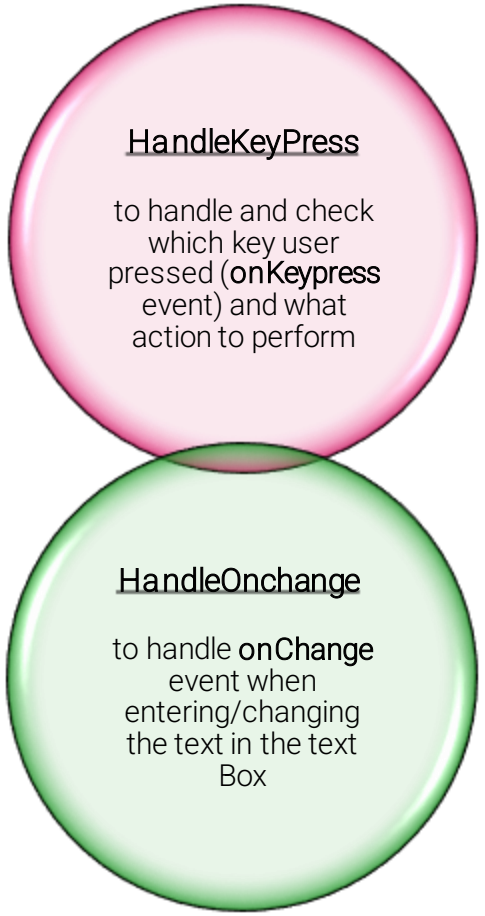
© Kindred Group plc 2021

Classified as General

React Events

```
handleKeyPress(event) {
    const key = event.key || event.keyCode || event.which;
    if (key === 'Enter' || key === 13 || key === 0) {
        this.postATweet();
    }
}
```

```
handleOnchange = (event) => {
    const {value} = event.target;

    this.setState({
        tweet: value,
        charCounter: value.length
    })
};
```


HandleKeyPress
to handle and check which key user pressed (onKeypress event) and what action to perform
HandleOnchange
to handle onChange event when entering/changing the text in the text Box

© Kindred Group plc 2021

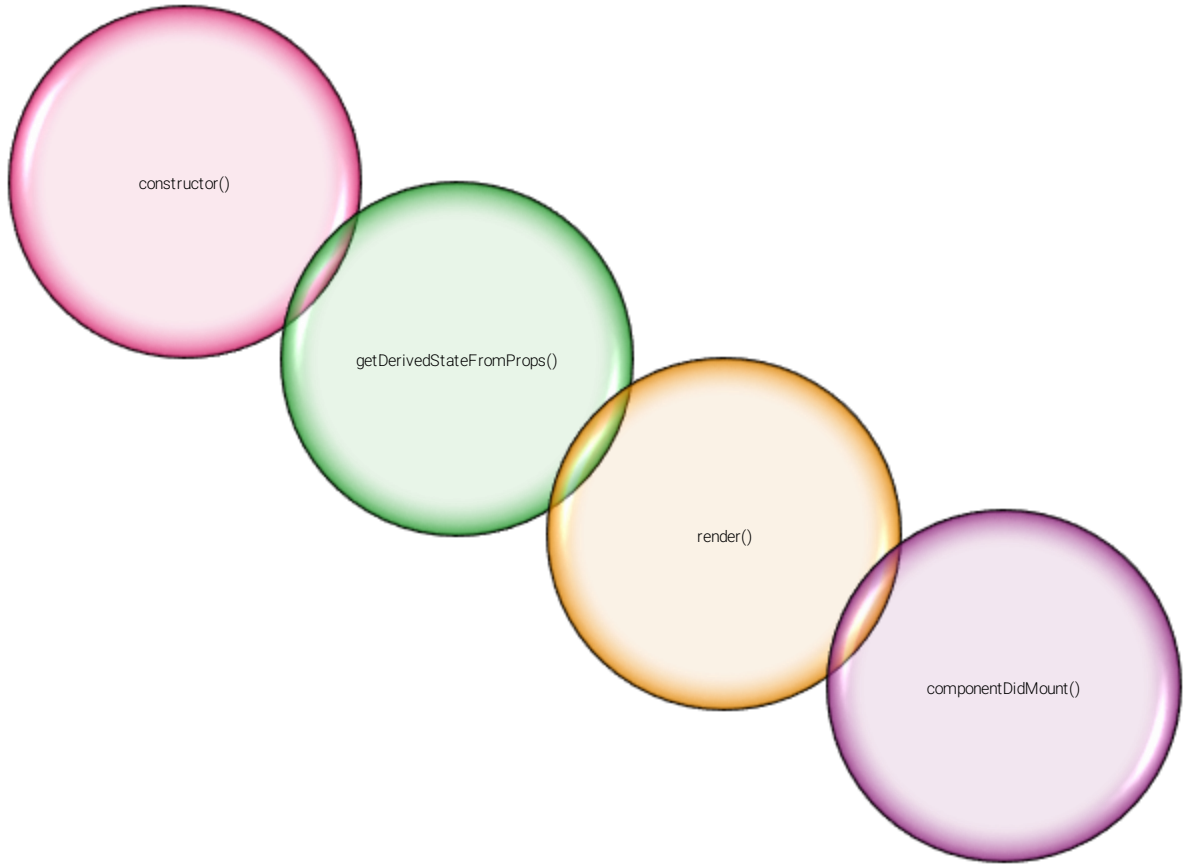
Classified as General

React LifeCycle: Mounting

Mounting means putting elements into the DOM.

React has four built-in methods that gets called, in this order, when mounting a component

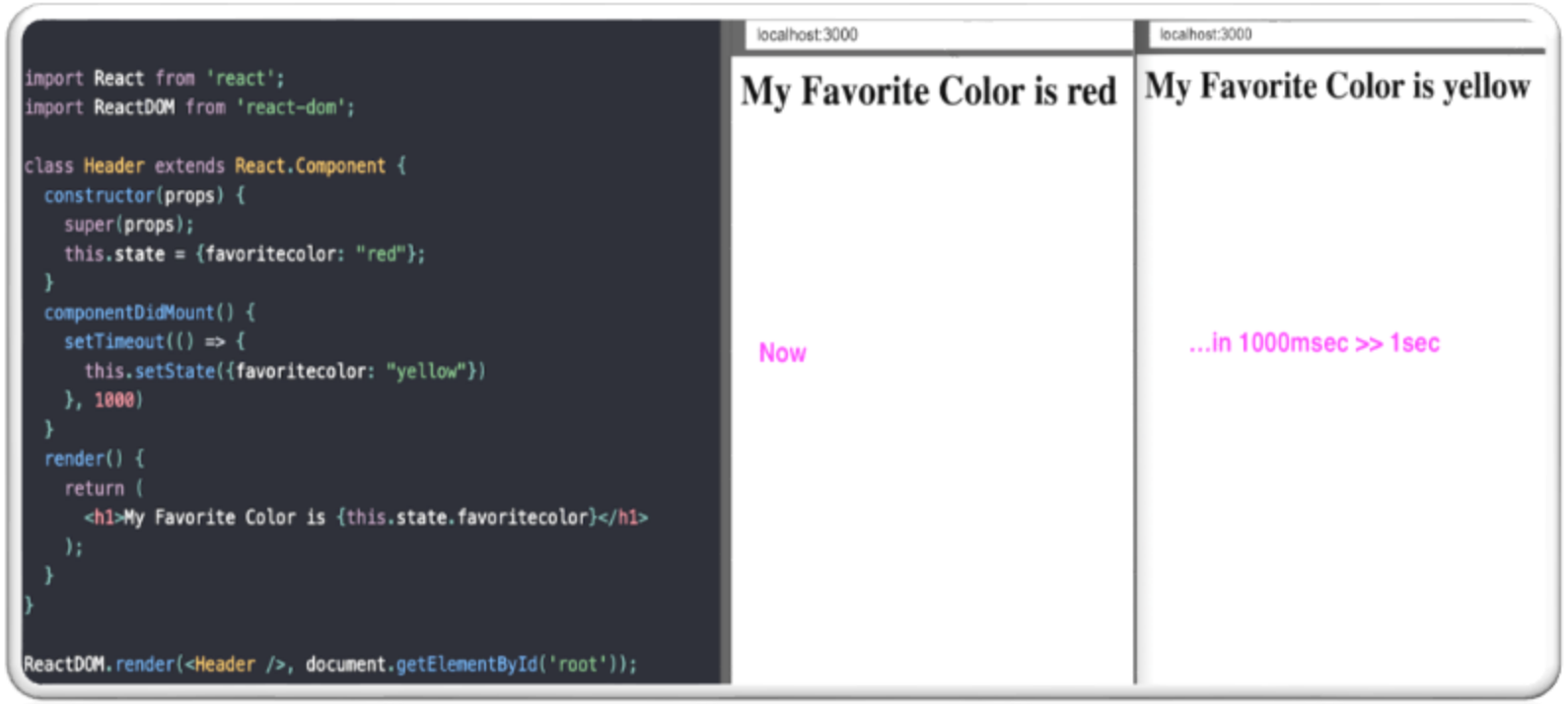
The render() method is required and will always be called, the others are optional and will be called if you define them.



© Kindred Group plc 2021

Classified as General

React LifeCycle: componentDidMount()



componentDidMount()

is invoked immediately after a component is mounted (inserted into the tree). Initialization that requires DOM nodes should go here. If you need to load data from a remote endpoint, this is a good place to instantiate the network request.

© Kindred Group plc 2021

Classified as General

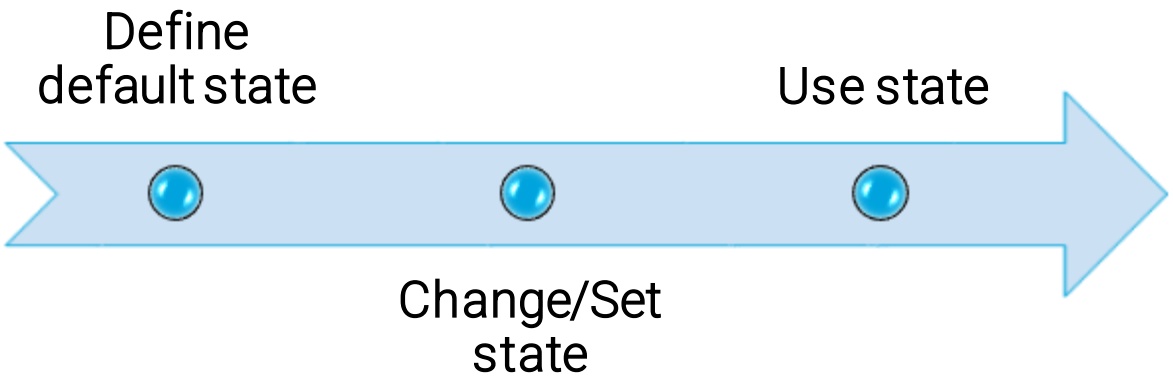
React state

```
//Define default state
state = {
  tweet: '',
  tweets: [],
  charCounter: 0,
  limit: 140
}

// Change/Set state
this.setState({
      tweet: value,
      charCounter: value.length
  })

// Use state
tweet={this.state.tweet}
```

React components has a built-in state object.

The state object is where you store property values that belongs to the component.

When the state object changes, the component re-renders.

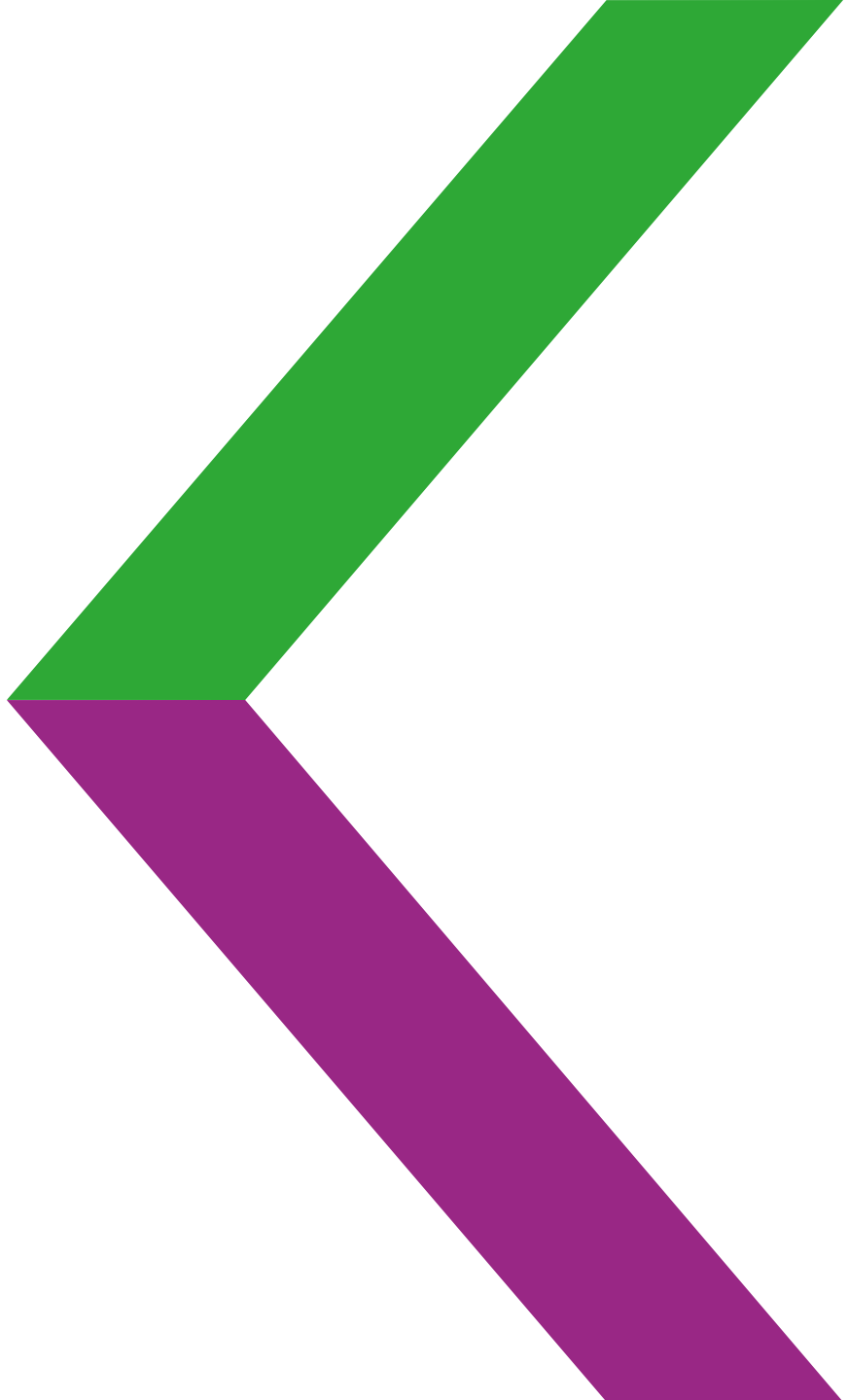


© Kindred Group plc 2021

Classified as General

Home assignment:

Use React to re-write your Survey Form from previous FE sessions



© Kindred Group plc 2021

Free resources to read and practice:

https://www.w3schools.com/js

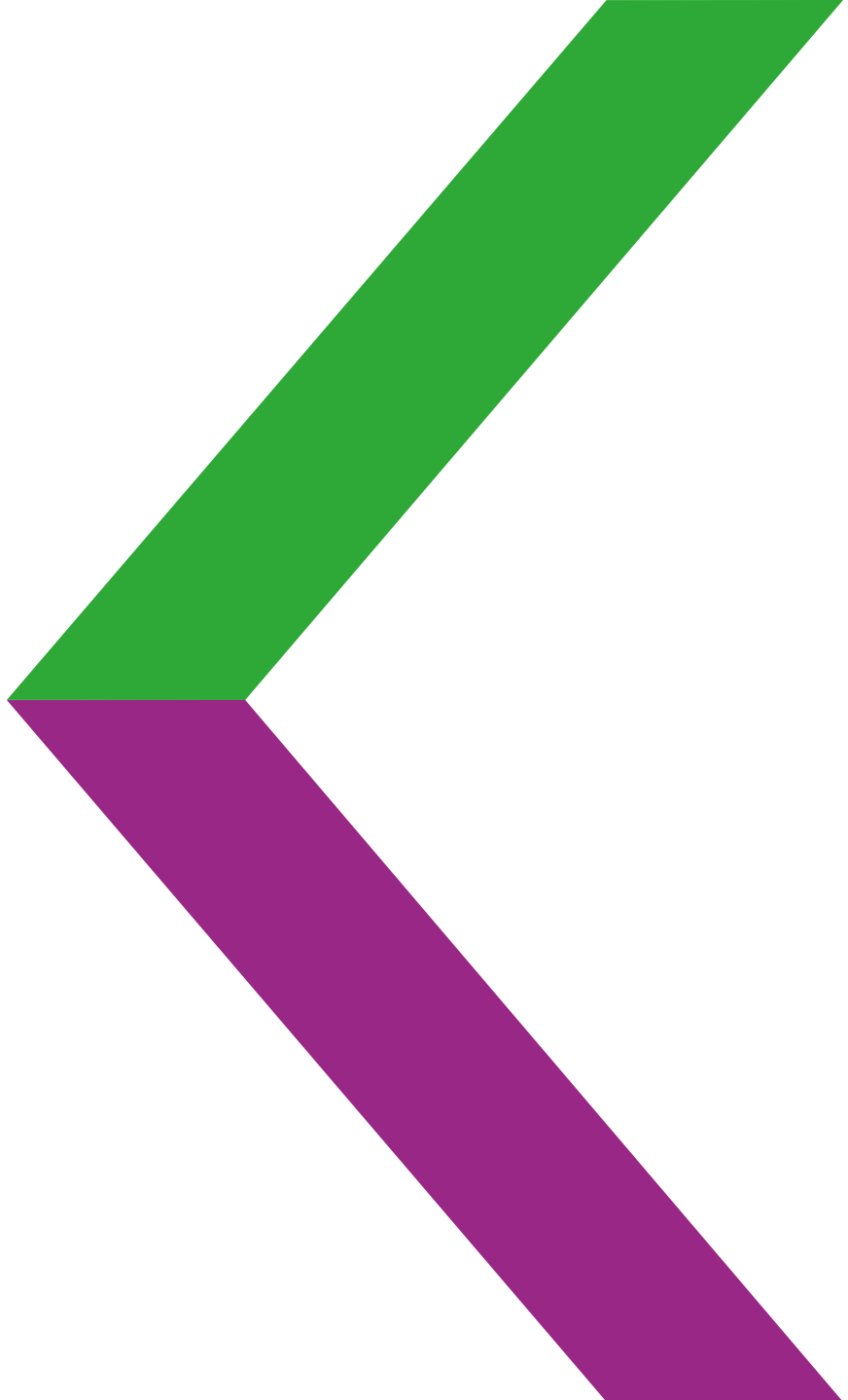
https://www.w3schools.com/REACT

https://reactjs.org/docs/getting-started.html

https://www.freecodecamp.org/learn

© Kindred Group plc 2018

Questions?



© Kindred Group plc 2021

Thank You

© Kindred Group plc 2021